J M S K
Jurnal Matematika, Statistika, & Komputasi
http://journal.unhas.ac.id/index.php/jmsk
Vol. 15, No. 1, 13-20, Juli 2018

p-ISSN: 1858-1382
e-ISSN: 2614-8811

Pengujian *Avalanche Weight Distribution* (AWD) *Criterion* pada Algoritma Serpent

Sandromedo Christa Nugroho
Badan Koordinasi Penanaman Modal*

Abstrak

Berdasarkan pada jenisnya, algoritma kriptografi dapat diklasifikasikan kedalam 3 jenis, yaitu algoritma asimetrik, algoritma simetrik dan algoritma fungsi hash. Teknik untuk mengukur level kekuatan dan keamanan pada algoritma simetrik, baik algoritma stream cipher maupun algoritma block cipher adalah dengan menggunakan serangan dan pengujian, salah satu teknik pengujian yang umum digunakan pada algoritma block cipher adalah Avalanche Weight Distribution (AWD) Criterion, yaitu sebaran histogram dari hamming weight vektor avalanche pada cipherteks yang merupakan kriteria untuk sebuah analisis sifat konfusi dan difusi pada suatu algoritma block cipher. Tulisan ini akan membahas mengenai pengujian AWD menggunakan 20.000 sample plainteks dengan 4 karakteristik kunci pada salah satu algoritma finalis kompetisi Advanced Encryption Standard (AES) yang diselengarakan oleh National Institute of Standards and Technology (NIST), Amerika Serikat, yaitu algoritma Serpent, serta melakukan perbandingan hasil pengujiannya dengan algoritm-algoritma block cipher finalis AES dan standar pada negara lainnya.

Kata Kunci: *Pengujian AWD, Algoritma Serpent*

1. Pendahuluan

Serangan dan pengujian merupakan salah dua teknik/metode yang dapat digunakan untuk mengukur level keaman dari sebuah algoritma *block cipher*. Serangan pada algoritma *block cipher* umumnya dilakukan dengan menggunakan karakteristik komponen dan struktur dari algoritma *block cipher* yang akan diserang, untuk kemudian dilakukan serangan dengan menggunakan *differential attack* atau *linear attack* dan serangan lainnya. Sedangkan pengujian pada algoritma *block cipher* umumnya dilakukan untuk mencari karakteristik komponen dan struktur pada algoritma *block cipher*, serta melakukan pengujian fungsi nonlinear secara black box, antara lain dengan menggunakan pengujian *Avalanche Criterion* (AC), *Strict Avalanche Criterion* (SAC), *Avalanche Weight Distribution* (AWD) *Criterion*, *Bit Independence Criterion* (BIC) dan pengujian lainnya. Pada tulisan ini akan dibahas mengenai pengujian AWD pada algoritma Serpent dengan menggunakan sampling untuk mewakili populasi, selain itu juga akan dilakukan perbandingan hasil pengujian AWD algoritma Serpent dengan algoritma *block cipher* lainnya, sebagai langkah awal dalam perbandingan level keamanan berdasarkan pada sifat konfusi dan difusi desain rancang bangun suatu algoritma *block cipher*, sehingga dapat menjadi

* major.ruft@gmail

informasi tambahan dan bahan pertimbangan dalam pemilihan implementasi algoritma *block cipher* untuk mengamankan informasi yang bersifat rahasia.

2. Algoritma Serpent

Ditemukannya serangan *differential attack* pada algoritma *Data Encryption Standard* (DES), panjang kunci yang tidak lagi optimal untuk mengamankan informasi yang bersifat rahasia, serta tuntutan publik untuk melakukan open competition algorithm project, menuntut *National Institute of Standards and Technology* (NIST) untuk melakukan sayembara dalam pemilihan algoritma *block cipher* yang akan dijadikan standar algoritma *block cipher* baru untuk mengantikan algoritma DES, yaitu algoritma *Advanced Encryption Standard* (AES). Sayembara tersebut dimulai pada tahun 1997 sampai dengan tahun 2001 dan diikuti oleh 15 algoritma *block cipher* yang dirancang bangun oleh beberapa negara dan/atau perkumpulan peneliti kriptografi dari kalangan akademisi. Pemilihan algoritma AES dilakukan tidak hanya berdasarkan pada keamanan semata, melainkan juga kinerja/performa, kebutuhan gate equivalent dan implementasi pada perangkat, khususnya perangkat dengan *source* yang terbatas. Pada tahun 1999, NIST mengumumkan terdapat 5 finalis yang menjadi kandidat kuat untuk menjadi algoritma AES, algoritma tersebut antara lain : algoritma MARS, algoritma RC6, algoritma Rijndael, algoritma Serpent dan algoritma Twofish. Kompetisi tersebut akhirnya dimenangkan oleh algoritma Rijndael karya Vincent Rijmen dan Joan Daemen, sedangkan pada posisi kedua dimenangkan oleh algoritma Serpent karya Ross Anderson, Eli Biham dan Lars Knudsen.

Algoritma Serpent memiliki ukuran blok data 128 bit dengan ukuran kunci yang bervariasi, yaitu 128, 192 dan 256 bit. Algoritma tersebut menggunakan struktur *Substitution Permutation Network* (SPN) dan membagi input blok data menjadi 4 yang masing-masing berukuran 32 bit, dimana setiap *round* pada algoritma Serpent menggunakan 1 dari 8 sbox 4 bit sebanyak 32 kali secara paralel. Secara keamanan 16 *round* pada algoritma Serpent dapat dikatagorikan aman dari serangan kriptanalisis yang mungkin dilakukan pada algoritma *block cipher* tersebut, namun algoritma Serpent didisain dengan menggunakan 32 *round* agar dapat bertahan dari serangan kriptanalisis lanjutan dimasa yang akan datang. Algoritma Serpent tidak dipatenkan ataupun masuk kedalam dokumen ataupun *domain* standar keamanan tertentu, sehingga dapat bebas digunakan oleh publik untuk mengamankan informasi yang bersifat rahasia. Tabel dibawah menunjukkan spesifikasi Algoritma Serpent.

Tabel Spesifikasi Algoritma Serpent.

No.	Keterangan	Deskripsi
1.	Pengembang/Pempublikasi	Ross Anderson, Eli Biham dan Lars Knudsen
2.	Tahun Publikasi	1998
3.	Ukuran Kunci	128 / 192 / 256 Bit
4.	Ukuran Block Plaintext/Ciphertext	128 Bit
5.	Struktur	SPN
6.	*Round*	32
7.	Dokumen Standar	---

3. Populasi dan Sampel Penelitian

Populasi adalah wilayah generalisasi yang terdiri atas obyek atau subyek yang mempunyai kuantitas dan karakteristik tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya (Sugiyono, 2003). Penelitian pada suatu populasi yang sangat besar dapat dilakukan dengan menggunakan sampel data yang mewakili populasi tersebut, dimana statistik yang diperoleh dari pengamatan terhadap sampel dapat digunakan sebagai bahan pengambilan kesimpulan atas populasi yang sedang diamati. Oleh karena itu, sampel yang digunakan didalam suatu penelitian harus dapat mewakili karakteristik suatu populasi. Terdapat berbagai teknik yang dapat digunakan untuk mengambil sampel yang akan digunakan dalam suatu penelitian. Menurut Dowdy, 2004 salah satu teknik yang paling sederhana adalah dengan menggunakan *simple random sampling*. Contoh N didefinisikan sebagai jumlah anggota populasi dan n sebagai jumlah anggota sampel. Jika pengambilan sampel dilakukan dengan suatu cara dimana setiap (N¦n) sampel memiliki peluang yang sama untuk dipilih, maka pengambilan sampel dikatakan sebagai *simple random sampling*.

4. *Avalanche Weight Distribution* (AWD) *Criterion*

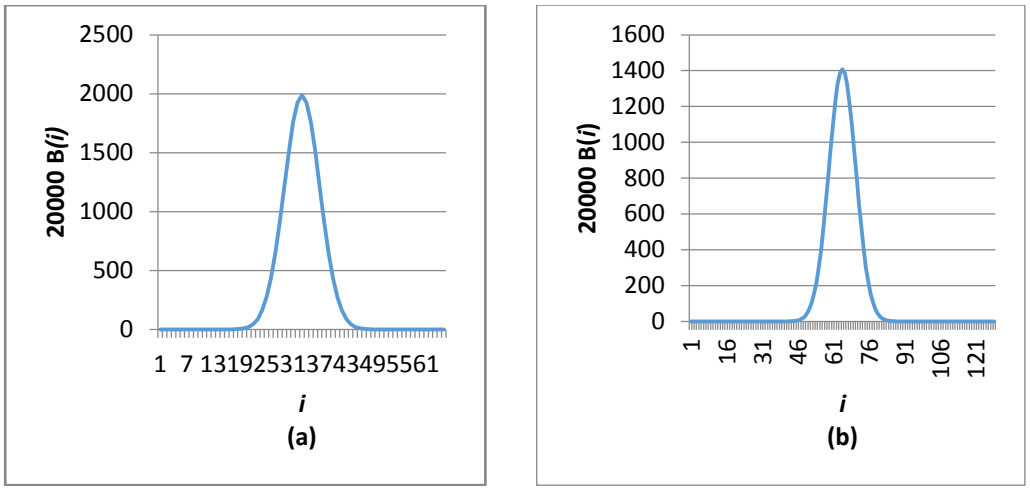
AWD pertama kali diperkenalkan oleh seorang ahli kriptografi bernama Ekrem Aras, dan Melek D. Yücel pada tahun 1999. AWD adalah histogram dari hamming weight vektor avalanche pada cipherteks, yaitu salah satu kriteria sederhana untuk sebuah analisis sifat konfusi dan difusi pada suatu algoritma *block cipher*. Vektor avalanche adalah banyaknya beda antara hasil cipherteks dari plainteks asli dengan hasil cipherteks dari plainteks yang telah diubah 1 bitnya, dimana pasangan plainteks (P1, P2) yang baik pada suatu algoritma *block cipher* memiliki histogram dari hamming weight vektor avalanche yang acak secara keseluruhan, sehingga kurva AWD yang berkaitan dengan seluruh pasangan yang mungkin untuk digunakan dari input yang hampir sama harus terdistribusi secara binomial di sekitar n/2 (n adalah panjang bit block data pada algoritma yang diuji). Algoritma *block cipher* yang memenuhi sifat difusi yang baik, idealnya akan memenuhi persamaan probabilitas sejumlah i perubahan bit dari n bit cipherteks :

$$B(i) = \frac{\binom{n}{i}}{2^n}, 0 \leq i \leq n$$

persamaan diatas merupakan persamaan binomial dimana

$$\sum_{i=0}^{n} B(i) = 1$$

Gambar dibawah menunjukkan kurva binomial ideal untuk n = 64 dan n = 128.



Gambar Kurva Distribusi Binomial Ideal Untuk (a) n= 64 dan (b) n= 128.

Salah satu tujuan pengujian AWD berfokus pada pengukuran dan perhitungan terhadap penyimpangan distribusi kurva AWD suatu algoritma *block cipher* terhadap kurva distribusi binomial ideal *B(j)* dengan menggunakan *N* pasangan plainteks, *difference* plainteks yang tetap ($\Delta P = P_1 \oplus P_2$), dan cipherteks yang berkorespondensi (C_1 dan C_2). Pengujian AWD dilakukan dengan menambah nilai elemen ke-*j array* AWD dengan nilai "1" berdasarkan hasil *difference* $\Delta C = C_1 \oplus C_2$ dengan *weight j*. Selanjutnya, ukuran penyimpangan (D^i) dan parameter *resemblance* (R^i) dapat dihitung dengan menggunakan prosedur sebagai berikut :

$$\mathrm{D}^i = \frac{1}{2N} \sum_{j=0}^{n} \left| AWD(j) - NB(j) \right|$$

dengan i berkorespondensi dengan posisi perubahan satu bit pada plainteks. Parameter resemblance Ri kemudian dihitung melalui persamaan dibawah :

$$\mathrm{R}^i = 1 - \mathrm{D}^i$$

Pada persamaan diatas nilai mutlak digunakan untuk tidak membedakan antara nilai *error* positif dan negatif, sedangkan koefisien normalisasi 1/2N digunakan untuk membatasi nilai Ri pada kasus terburuk yaitu Ri = 0. Jika Ri = 1, maka AWD algoritma *block cipher* yang diuji sama dengan distribusi binomial ideal yang diharapkan. Sebaliknya, jika nilai Ri = 0, maka AWD algoritma *block cipher* yang diuji tidak menunjukkan kemiripan dengan distribusi binomial ideal atau dalam hal ini sifat konfusi dan difusi algoritma *block cipher* tersebut tidak baik. Adapun langkah-langkah pada pengujian AWD, antara lain :

a. Plainteks = Bangkitkan 1 (satu) buah plainteks sesuai ukuran block size algoritma *block cipher*nya (64 bit, 128 bit, 192 bit, 256 bit, dan lain-lain) secara *random*;
b. Cipherteks = Lakukan enkripsi terhadap plainteks pada langkah ke-1;
c. Plainteks-Flip = Flip/ubah 1 (satu) bit pada plainteks langkah ke-1 pada posisi ke-1 sampai dengan posisi ke-n (contoh 64 bit / 128 bit / 256 bit);
d. Cipherteks- Flip = Lakukan enkripsi terhadap Plainteks- Flip pada langkah ke-3;
e. Difference = Hitung difference dengan cara melakukan XOR antara Cipherteks dengan Cipherteks-Flip;
f. Hamming Weight = Hitung hamming weight vektor pada Difference langkah ke-5;
    1) Contoh :
        a) Block size yang akan diuji = 128 bit;

   b) Hasil perhitungan Hamming Weight = 64.

g. Tabel = Masukkan hasil perhitungan Hamming Weight pada sebuah tabel array AWD (dimana Tabel akan berukuran sama dengan block size algoritma *block cipher* + 1, karena nilai "0" juga dihitung);
   1) Contoh :
      a) Hasil perhitungan Hamming Weight = 64;
      b) Maka tabel perubahan bit ke-1 posisi ke-64 bertambah 1.
h. Tabel Rekursif = Lakukan langkah ke-a sampai dengan ke-g sebanyak N kali;
   1) Contoh :
      a) N = 10.000 kali (menurut Agus Buono, 2016 secara statistik 10.000 sampel plainteks sudah cukup untuk mewakili populasi, selain itu 10.000 sampel juga digunakan pada tesis Savas Arikan mengenai pengujian AWD pada algoritma *block cipher*);
      b) N = 1.048.576 kali (menurut hasil pembahasan tim rancang bangun alat uji algoritma *block cipher* tahun anggaran 2016)
      c) Dalam hal ini posisi :
         i. Jumlah ke-0 sampai dengan jumlah ke-65 akan berjumlah 10.000 / 1.048.576 (untuk algoritma dengan block size 64 bit);
         ii. Jumlah ke-0 sampai dengan jumlah ke-129 akan berjumlah 10.000 / 1.048.576 (untuk algoritma dengan block size 128 bit).
      d) Sedangkan
         i. Hasil yang baik adalah pada posisi jumlah ke-32 berjumlah 5.000 / 524.288, serta menyebar merata membentuk kurva distribusi normal (untuk algoritma dengan block size 64 bit);
         ii. Hasil yang baik adalah pada posisi jumlah ke-64 berjumlah 5.000 / 524.288, serta menyebar merata membentuk kurva distribusi normal (untuk algoritma dengan block size 128 bit).
i. Tabel Hasil = Lakukan penyesuaian pada setiap perubahan bit antara Tabel Rekursif pada langkah ke-8 dengan Tabel Binomial Ideal
   1) Contoh :
      a) Perubahan bit ke-1 posisi ke-1 adalah 0
      b) Perubahan bit ke-1 posisi ke-2 adalah 0 ...
      c) Perubahan bit ke-1 posisi ke-64 adalah (5.000 / 524.288)
      d) Perubahan bit ke-1 posisi ke-65 adalah (5.000 / 524.288) – x ...
      e) Perubahan bit ke-1 posisi ke-128 adalah 0
      f) Perubahan bit ke-1 posisi ke-129 adalah 0
j. Kurva = Nilai-nilai Tabel Hasil pada pengujian AWD yang baik akan membentuk kurva Distribusi Normal untuk setiap perubahan bit
   1) Catatan :
      a) Perubahan bit untuk block size 64 bit adalah 64 perubahan
      b) Perubahan bit untuk block size 128 bit adalah 128 perubahan
k. Deviasi = Lakukan perhitungan nilai deviasi untuk setiap perubahan bit pada table
l. *Resemblance* = Lakukan perhitungan nilai resemblance untuk setiap perubahan bit berdasarkan nilai deviasi pada langkah k
m. Hasil = Nilai *error* maksimal pada pengujian AWD yang semakin mendekati nilai satu, akan semakin baik

5. Asumsi Pengujian AWD pada Algoritma Serpent

Pengujian algoritma Serpent pada tulisan ini dilakukan dengan menggunakan 4 buah kunci sepanjang 128 bit yang berbeda-beda karakteristiknya, yaitu *low density*, *high density*, *random*, *secure random* versi bahasa pemrograman Java. Tabel dibawah menunjukkan kunci 128 bit yang akan digunakan dalam pengujian AWD.

Tabel Kunci 128 Bit.

No.	**Input Kunci**
Low density	
1.	0100 0000 0000 0000 0000 0000 0000 0000
2.	0020 0000 0000 0000 0000 0000 0000 0000
3.	0000 4000 0000 0000 0000 0000 0000 0000
4.	0000 0008 0000 0000 0000 0000 0000 0000
5.	0000 0000 0010 0000 0000 0000 0000 0000
6.	0000 0000 0000 2000 0000 0000 0000 0000
7.	0000 0000 0000 0004 0000 0000 0000 0000
8.	0000 0000 0000 0800 0000 0000 0000 0000
9.	0000 0000 0010 0000 0000 0000 0000 0000
10.	0000 0002 0000 0000 0000 0000 0000 0000
High density	
1.	FEFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
2.	FFDF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
3.	FFFF BFFFF FFFF FFFF FFFF FFFF FFFF FFFF
4.	FFFF FFF7 FFFF FFFF FFFF FFFF FFFF FFFF
5.	FFFF FFFF FEFF FFFF FFFF FFFF FFFF FFFF
6.	FFFF FFFF FFFD FFFF FFFF FFFF FFFF FFFF
7.	FFFF FFFF FFFF FFBF FFFF FFFF FFFF FFFF
8.	FFFF FFFF FFF7 FFFF FFFF FFFF FFFF FFFF
9.	FFFF FFFF EFFF FFFF FFFF FFFF FFFF FFFF
10.	FFFF FDFF FFFF FFFF FFFF FFFF FFFF FFFF
Random	
1.	C68A 331C 1287 03DC 72AC 8E77 1905 019E
2.	66D3 23F3 3CE0 191E A3D9 1946 3BE0 32B2
3.	9ABA 5CE5 24C5 1FF2 1027 CF9F FDC1 6F22
4.	A65B 44F7 EDA5 4A49 B235 FB20 2874 F482
5.	4656 AA8F C6EF 7DB0 D1CF C8E4 068E 33BB
6.	437D 7551 B509 37C7 0289 5150 3F12 9877
7.	AD97 1C3F 6A2E 6D10 C997 607A A9F7 BBBA
8.	F07C 7F7D 63B9 A474 A168 395F CD1B CA7C
9.	A044 A5CE 9CCE 7411 6346 291A 91D7 8EC1
10.	FE03 A5FE 2A1E 534A 8F79 41AC DC52 BE45
Secure random	
1.	3618 20F1 A8CE 5A0A BE45 12E5 5BF9 2F60
2.	F8B9 A65A 83A8 37A6 C8EC F45E F3B7 6006
3.	2168 AEBD 5042 0EFD B0F2 089B 68E3 B70B
4.	5F8F 7F56 57B5 6C3B 6AD5 E907 6CAE 002F
5.	3626 C71B CD43 DEBA DFF0 57B8 1823 ACD4
6.	5C3C D2EA 99F2 5FB4 4244 A809 0AAA 88AF

7.	6379 B3A0 4FC8 38BC D493 6B5F 5F3B EBB4
8.	CB24 0666 A541 C8B7 4B89 5ECB 8FD0 AA6E
9.	5EA1 EC0D 3745 A3B1 32AC 10C0 F8B8 F0DF
10.	95DB 0371 9495 56CA 4914 C302 4A78 A4BA

6. Hasil Pengujian AWD

Pengujian AWD pada tulisan ini akan dilakukan dengan menggunakan 20.000 sampel plainteks untuk mewakili populasi dari keseluruhan kemungkinan plainteks, sedangkan kunci yang akan digunakan adalah kunci dengan karakteristik *low density*, *high density*, *random* dan *secure random* sebanyak 10 buah. Pengujian dilakukan pada masing-masing kunci untuk mencari tabel AWD, kemudian dicari nilai deviasi pada masing-masing perubahan bit, lalu dicari nilai resemblance dan *error*nya. Nilai *error* maksimal resemblance hasil pengujian akan digunakan sebagai nilai pembanding hasil pengujian AWD algoritma Serpent dengan algoritma *block cipher* lainnya. Tabel dibawah menunjukkan hasil pengujian AWD algoritma Serpent.

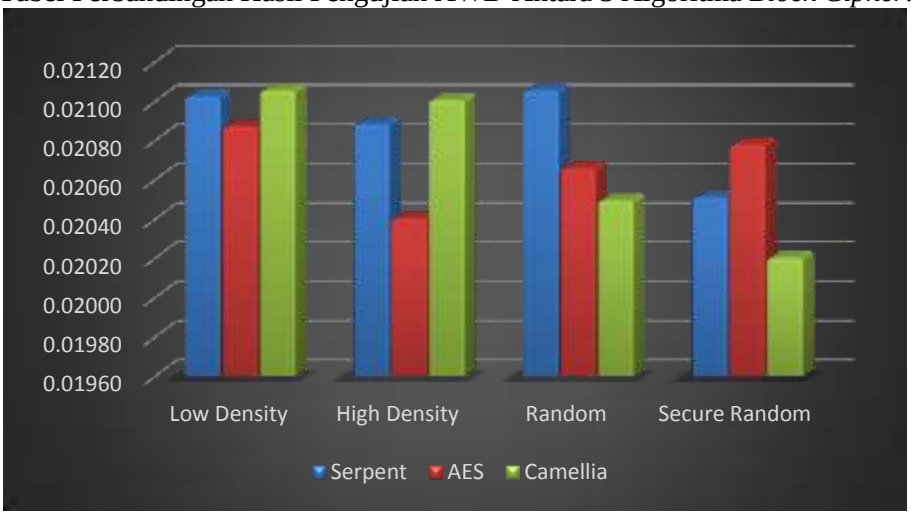
Tabel Hasil Pengujian AWD Algoritma Serpent.

No.	**Informasi**	***Resemblance* Maksimal**	***Resemblance* Minimal**	**Rata-Rata *Resemblance***	***Error* Maksimal *Resemblance***
Low density					
1.	Pengujian 1	0.98963	0.97820	0.98563	0.02180
2.	Pengujian 2	0.98932	0.97771	0.98512	0.02229
3.	Pengujian 3	0.99080	0.98009	0.98533	0.01991
4.	Pengujian 4	0.99022	0.97682	0.98498	0.02318
5.	Pengujian 5	0.98989	0.97975	0.98535	0.02025
6.	Pengujian 6	0.99074	0.97848	0.98506	0.02152
7.	Pengujian 7	0.99195	0.98007	0.98519	0.01993
8.	Pengujian 8	0.99001	0.98025	0.98536	0.01975
9.	Pengujian 9	0.99035	0.97872	0.98522	0.02128
10.	Pengujian 10	0.99212	0.97964	0.98528	0.02036
Rata-Rata Nilai *Error Resemblance*					**0.02103**
High density					
1.	Pengujian 1	0.99092	0.97822	0.98540	0.02178
2.	Pengujian 2	0.99097	0.97843	0.98509	0.02157
3.	Pengujian 3	0.99063	0.98074	0.98533	0.01926
4.	Pengujian 4	0.99039	0.97823	0.98518	0.02177
5.	Pengujian 5	0.98971	0.97853	0.98533	0.02147
6.	Pengujian 6	0.99003	0.97943	0.98519	0.02057
7.	Pengujian 7	0.99065	0.97960	0.98558	0.02040
8.	Pengujian 8	0.99032	0.98004	0.98563	0.01996
9.	Pengujian 9	0.99059	0.97957	0.98526	0.02043
10.	Pengujian 10	0.99172	0.97835	0.98513	0.02165
Rata-Rata Nilai *Error Resemblance*					**0.02089**
Random					
1.	Pengujian 1	0.98962	0.97808	0.98548	0.02192
2.	Pengujian 2	0.98982	0.97783	0.98539	0.02217
3.	Pengujian 3	0.98920	0.97902	0.98497	0.02098
4.	Pengujian 4	0.98982	0.97955	0.98529	0.02045
5.	Pengujian 5	0.99093	0.97926	0.98543	0.02074
6.	Pengujian 6	0.99001	0.97902	0.98536	0.02098
7.	Pengujian 7	0.98983	0.97893	0.98532	0.02107
8.	Pengujian 8	0.99084	0.98061	0.98562	0.01939
9.	Pengujian 9	0.99045	0.97907	0.98532	0.02093
10.	Pengujian 10	0.98982	0.97804	0.98514	0.02196

Rata-Rata Nilai *Error Resemblance*					**0.02106**
Secure random					
1.	Pengujian 1	0.99065	0.97945	0.98541	0.02055
2.	Pengujian 2	0.99006	0.97977	0.98534	0.02023
3.	Pengujian 3	0.99075	0.98021	0.98502	0.01979
4.	Pengujian 4	0.98924	0.97985	0.98529	0.02015
5.	Pengujian 5	0.98905	0.97801	0.98538	0.02199
6.	Pengujian 6	0.98993	0.97888	0.98540	0.02112
7.	Pengujian 7	0.98896	0.98023	0.98528	0.01977
8.	Pengujian 8	0.98947	0.98007	0.98542	0.01993
9.	Pengujian 9	0.99241	0.97894	0.98526	0.02106
10.	Pengujian 10	0.99068	0.97947	0.98547	0.02053
Rata-Rata Nilai *Error Resemblance*					**0.02051**

Perbandingan hasil pengujian AWD pada algoritma Serpent akan dilakukan dengan 2 algoritma *block cipher* standar yang umum digunakan oleh khalayak umum, yaitu algoritma AES yang merupakan standar algoritma *block cipher* Amerika Serikat berdasarkan pada dokumen NIST FIPS 197 dan algoritma Camellia yang merupakan rekomendasi algoritma *block cipher* Jepang berdasarkan pada dokumen CRYPTREC dan NESSIE. Tabel dibawah menunjukkan perbandingan rata-rata *error* pada hasil pengujian AWD antara 3 algoritma *block cipher*.

Tabel Perbandingan Hasil Pengujian AWD Antara 3 Algoritma *Block Cipher*.



Berdasarkan pada hasil perbandingan hasil pengujian AWD antara 3 algoritma *block cipher* diatas, dapat diketahui bahwa, algoritma AES ungul pada 3 karakteristik kunci, yaitu *low density*, *high density* dan *random*, sedangkan algoritma Camellia ungul pada 1 karakteristik kunci, yaitu *secure random.*

7. Kesimpulan

Berdasarkan pada hasil pengujian AWD algoritma Serpent diatas dapat diambil beberapa kesimpulan, antara lain :

a. Nilai minimal *resemblance* hasil pengujian AWD pada algoritma Serpent dengan menggunakan kunci berkarakteristik *low density*, *high density*, *random* dan *secure random* adalah sangat baik, dalam hal ini mendekati nilai 1/sempurna, yaitu berkisar

antara 0.976 sampai dengan 0. 980 dan nilai rata-rata *resemblance*nya adalah berkisar antara 0.984 sampai dengan 0.985 dengan error maksimalnya adalah 0.02318.

b. Berdasarkan pada hasil pengujian AWD diatas dan dengan menggunakan nilai *error* bound (alfa) 0.05, maka dapat dinayatakan bahwa algoritma Serpent lulus pengujian AWD.
c. Hasil perbandingan hasil pemngujian AWD antara algoritma Serpent, algoritma AES dan algoritma Camellia menunjukkan bahwa algoritma AES memiliki hasil pengujian AWD yang lebih baik dibandingkan dengan kedua algoritma *block cipher* lainnya. Namun perlu diketahui perbandingan tersebut bukan berarti menyatakan algoritma AES lebih kuat dan aman, karena pengujian AWD hanya merupakan salah satu pengujian algoritma *block cipher*, dimana masih terdapat beberapa pengujian dan serangan lainnya yang perlu dilakukan untuk benar-benar membandingkan level keamanan satu algoritma dengan algoritma *block cipher* lainnya.

Daftar Pustaka

[1] Arikan, Savas. 2003. *Propagation Characteristics of RC5, RC6, and Twofish Ciphers*. Thesis. School of Natural and Applied Sciences of Middle Esat Technical University.

[2] Kavut, Selcut. Yucel D Melek. *On Some Cryptographic Properties of Rijndael*. Middle East Technical University and Information Technologies and Electronic Research Institute.

[3] *National Institute of Standards and Technology* (NIST). *Federal Information Processing Standart Publication* (FIPS) 197. 2001. *Advanced Encryption Standart (AES)*.

[4] Soto Juan. 1999. *Randomness Testing of the Advance Encryption Standard Candidate Algorithms.* US Department of Commerce. Technology Administration.

[5] Sulak, Fatih. 2011. *Statistical Analysis of Block ciphers and Hash Functions*. Thesis. Middle East Technical University.

[6] Webster. Tavares. *On the Design of S-boxes*. Queen's University. Kingston. Canada.